Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина: Архитектура вычислительных систем

*К защите допустить:*

И.О. Заведующего кафедрой информатики

*\_\_\_\_\_\_\_\_\_\_\_\_\_*С. И. Сиротко

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

на тему

**АВТОМАТИЧЕСКАЯ СИСТЕМА ПОЛИВА С ДАТЧИКАМИ ВЛАЖНОСТИ ПОЧВЫ НА *ARDUINO* *NANO***

БГУИР КП 1-40 04 01 001 ПЗ

Студент И. С. Авдошко

Руководитель А. А. Калиновская

Нормоконтролер А. А. Калиновская

Минск 2024

**СОДЕРЖАНИЕ**

[Введение 5](#_Toc184673228)

[1 Архитектура вычислительной системы 7](#_Toc184673229)

[1.1 Структура и архитектура вычислительной системы 7](#_Toc184673230)

[1.2 Архитектура *AVR* 8](#_Toc184673231)

[1.3 Датчики влажности почвы 10](#_Toc184673232)

[1.4 Насос для автоматической системы полива 11](#_Toc184673233)

[1.5 Дисплей *LCD*1602 с *I2C* адаптером 12](#_Toc184673234)

[1.6 Модуль реле 14](#_Toc184673235)

[2 Платформа программного обеспечения 16](#_Toc184673236)

[2.1 История операционной системы *Windows* 16](#_Toc184673237)

[2.2 Преимущества операционной системы *Windows* 18](#_Toc184673238)

[2.3 *Arduino IDE* 18](#_Toc184673239)

[3 Теоретическое обоснование разработки программного продукта 20](#_Toc184673240)

[3.1 Обоснование необходимости разработки 20](#_Toc184673241)

[3.2 Технологии программирования, используемые для решения поставленных задач 21](#_Toc184673242)

[3.3 Связь архитектуры вычислительной системы с разрабатываемым прибором 23](#_Toc184673243)

[3.4 Актуальность разработки системы 24](#_Toc184673244)

[4 Проектирование функциональных возможностей программы 26](#_Toc184673245)

[5 Архитектура разрабатываемой программы 28](#_Toc184673246)

[5.1 Общая структура системы автоматического полива 28](#_Toc184673247)

[5.2 Описание функциональной схемы программы 32](#_Toc184673248)

[5.3 Описание блок-схемы алгоритма программы 36](#_Toc184673249)

[Заключение 40](#_Toc184673250)

[Список литературных источников 41](#_Toc184673251)

[Приложение А (обязательное) Справка о проверке на заимствования 43](#_Toc184673252)

[Приложение Б (обязательное) Листинг программного кода 44](#_Toc184673253)

[Приложение В (обязательное) Функциональная схема алгоритма, реализующего программное средство 48](#_Toc184673254)

[Приложение Г (обязательное) Блок-схема алгоритма, реализующего программное средство 49](#_Toc184673255)

[Приложение Д (обязательное) Графический интерфейс пользователя 50](#_Toc184673256)

[Приложение Е (обязательное) Ведомость курсового проекта 51](#_Toc184673257)

# ВВЕДЕНИЕ

В условиях современного городского образа жизни автоматизация процессов ухода за комнатными растениями становится особенно актуальной. Одной из важнейших задач является поддержание оптимального уровня влажности почвы, который обеспечивает здоровый рост растений. Однако уход за растениями, особенно в условиях ограниченного времени или отсутствия владельцев, становится трудоёмким процессом. Решением этой проблемы может стать система автоматического полива, которая самостоятельно контролирует влажность почвы и осуществляет полив по мере необходимости.

Актуальность темы курсового проекта обусловлена необходимостью повышения удобства ухода за комнатными растениями и оптимизации использования воды. Комнатные растения часто сталкиваются с пересушиванием или, наоборот, с чрезмерным поливом, что негативно влияет на их рост. Автоматические системы полива, базирующиеся на современных микроконтроллерах, таких как *Arduino Nano*, могут минимизировать человеческий фактор и обеспечить стабильный уход за растениями.

Целью данного курсового проекта является разработка и анализ архитектуры автоматической системы полива для комнатных растений с использованием датчиков влажности почвы на базе платформы *Arduino Nano*, а также исследование взаимодействия системы с её аппаратной частью и обработка данных в реальном времени. Особое внимание уделяется анализу краевых случаев, таких как аномальные значения с одного из датчиков или выход датчиков из строя.

Для достижения поставленной цели необходимо решить следующие задачи:  
 1 Изучить архитектуру вычислительной платформы *Arduino Nano*, включая её возможности по обработке данных с нескольких датчиков и взаимодействие с периферийными устройствами.

2 Проанализировать работу датчиков влажности почвы в условиях комнатных растений, исследовать их взаимодействие с микроконтроллером и возможные ошибки при измерениях.

3 Разработать алгоритмы сбора, обработки и анализа данных с датчиков, способные учитывать аномальные значения или сбой работы одного из датчиков.

4 Исследовать методы обработки краевых случаев, включая корректное поведение системы при выходе одного из датчиков из строя и устранение влияния ложных данных.

5 Реализовать программное решение для автоматической системы полива, интегрирующее контроль за состоянием датчиков и управление процессом полива на базе платформы *Arduino*.

6 Оценить работу системы в различных режимах, проанализировать её устойчивость и надежность при появлении неисправностей в датчиках.

Таким образом, данная работа направлена на комплексное исследование архитектуры вычислительной системы на платформе Arduino Nano с целью создания эффективной автоматической системы полива для комнатных растений, способной корректно реагировать на аномальные ситуации и обеспечивать стабильную работу в различных условиях. Эта система будет не только минимизировать необходимость в постоянном внимании со стороны владельцев, но и гарантировать оптимальное состояние растений, что особенно важно в условиях городской жизни. В конечном итоге, разработанная система будет способствовать более рациональному использованию ресурсов и обеспечит устойчивый подход к уходу за растениями, улучшая их жизнеспособность и декоративность.

Пояснительная записка оформлена в соответствии с СТП 01-2024 [1].

# 1 АРХИТЕКТУРА ВЫЧИСЛИТЕЛЬНОЙ СИСТЕМЫ

## 1.1 Структура и архитектура вычислительной системы

Микроконтроллер – это компактное вычислительное устройство, предназначенное для управления различными электронными системами и процессами. Он объединяет в себе процессор, память и порты ввода-вывода, что делает его ключевым компонентом в системах автоматизации и управления. Микроконтроллеры широко используются в различных сферах: от бытовой электроники до сложных промышленных систем. Благодаря своей универсальности и простоте, микроконтроллеры являются основой для построения умных устройств и систем, требующих автономного управления.

Одним из таких микроконтроллеров является *Arduino Nano*, который представляет собой миниатюрную версию популярной платформы *Arduino*. Этот микроконтроллер основан на процессоре *ATmega328* и обладает множеством возможностей, позволяющих использовать его в самых разных проектах. Он поддерживает 14 цифровых входов и выходов, из которых 6 могут работать в режиме широтно-импульсной модуляции (ШИМ), а также имеет 8 аналоговых входов для подключения датчиков. Объем флэш-памяти составляет 32 кБ, что достаточно для размещения программы, управляющей устройством. Оперативная память (*SRAM*) – 2 кБ, а постоянная память (*EEPROM*) – 1 кБ. Все это работает с тактовой частотой 16 МГц, что обеспечивает выполнение базовых задач с низким энергопотреблением [2]. Схема микроконтроллера представлена на рисунке 1.1.

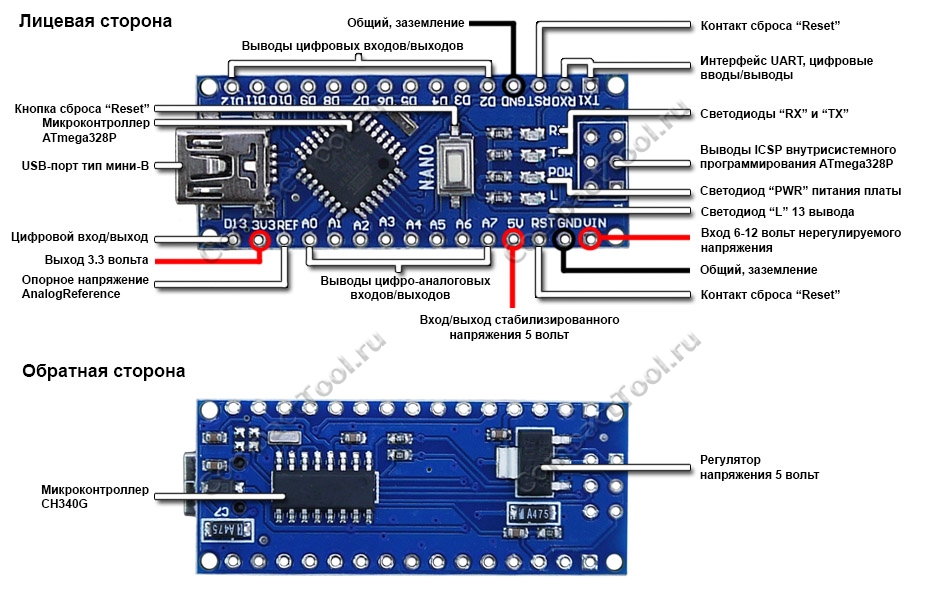


Рисунок 1.1 – Схема микроконтроллера *Arduino Nano*

Архитектура *Arduino Nano* построена на 8-битной архитектуре *AVR*, что делает его оптимальным для простых проектов, таких как системы автоматического управления, например, система полива [3]. Эта плата – часть более широкой экосистемы *Arduino*, которая была разработана в 2005 году для образовательных целей, чтобы облегчить студентам и энтузиастам создание электронных проектов. В зависимости от задач, существует несколько версий *Arduino*: от базовой *Arduino Uno* до более мощной *Arduino Mega*. Микроконтроллер *Arduino Nano* – это миниатюрная версия *Uno*, которая идеально подходит для проектов с ограниченным пространством [4].

Одним из ключевых преимуществ *Arduino Nano* является его компактность, которая позволяет интегрировать плату в небольшие системы. Это особенно важно для проектов, таких как автоматические системы полива растений, где пространство для установки ограничено. Кроме того, простой интерфейс программирования и наличие множества библиотек делает работу с платой доступной как для новичков, так и для опытных разработчиков. Стоимость устройства также является важным фактором – *Arduino Nano* является доступным решением для множества проектов.

Для создания системы автоматического полива выбор *Arduino Nano* был продиктован его небольшими размерами, достаточным количеством входов/выходов для подключения нескольких датчиков и исполнительных устройств (например, насосов и клапанов), а также низким энергопотреблением, что критично для автономных систем. Возможность легко настроить программное обеспечение и управлять процессами через встроенные таймеры делает эту плату идеальной для подобного рода приложений. Например, с помощью таймеров можно организовать полив в зависимости от показаний датчиков влажности почвы, что обеспечивает эффективное управление без излишних затрат ресурсов.

Важно учитывать, что несмотря на все достоинства, у *Arduino Nano* есть ограничения по памяти и вычислительной мощности, что требует оптимизации кода для эффективного управления системой. Однако для простых автоматизированных решений, таких как системы полива, эти ресурсы достаточны для выполнения всех необходимых функций.

## 1.2 Архитектура *AVR*

*AVR* – это архитектура микроконтроллеров, разработанная компанией *Atmel* (сейчас *Microchip*) в 1996 году. Она представляет собой 8-битную *RISC*-архитектуру (*Reduced Instruction Set Computing*), что делает её подходящей для простых и энергоэффективных проектов. Основная идея *AVR* заключается в минимизации количества машинных команд, что позволяет выполнять каждую команду за один или два такта. Это повышает эффективность и снижает энергопотребление, что особенно важно для маломощных систем [5].

Микроконтроллеры *AVR*, такие как *ATmega328*, используемый в *Arduino Nano*, обладают рядом особенностей, которые делают их популярными в любительских и профессиональных проектах. Они имеют встроенные модули для работы с аналоговыми и цифровыми сигналами, поддерживают интерфейсы передачи данных, такие как *SPI*, *I2C* и *UART*, а также способны обрабатывать сигналы широтно-импульсной модуляции (ШИМ). Это позволяет эффективно управлять мощностью в различных приложениях, включая двигатели и светодиоды.

Программная модель микроконтроллеров *AVR* включает в себя регистры общего назначения (РОН), регистры ввода-вывода, память программ, оперативную память данных и энергонезависимую память данных. РОН (*R*0…*R*31) могут использоваться в программе для хранения данных, адресов, констант и другой информации. Шесть старших регистров объединены попарно, формируя три 16-разрядных регистра – *X*[*R27*], *Y*[*R29*] и *Z*[*R31*][6]. Это упрощает операции с данными и адресами, повышая общую эффективность программирования.

Важным аспектом архитектуры *AVR* является использование кэш-памяти, что ускоряет работу с данными, хотя объёмы памяти здесь значительно меньше, чем в современных архитектурах, таких как *Alder Lake*. Архитектура поддерживает до 32 регистров общего назначения, которые связаны с арифметико-логическим устройством (*ALU*), и все регистры могут использоваться для операций чтения, записи и вычислений [7].

Также стоит отметить встроенные таймеры и прерывания, которые позволяют микроконтроллеру работать в реальном времени. Эти функции дают возможность выполнять задачи по расписанию или реагировать на внешние события. Программные прерывания позволяют управлять несколькими задачами параллельно, что делает *AVR* особенно эффективным для управления датчиками и исполнительными механизмами, например, в автоматических системах полива. Структурная схема *AVR* контроллера представлена на рисунке 1.2.

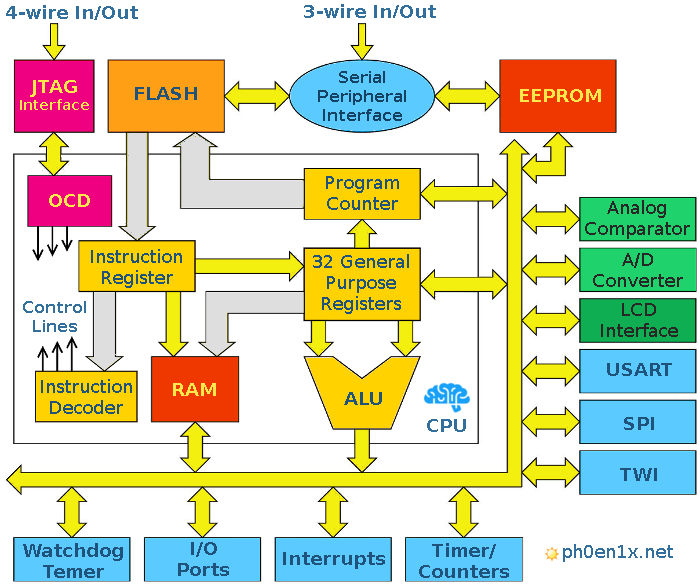


Рисунок 1.2 – Структурная схема *AVR* контроллера

## 1.3 Датчики влажности почвы

Датчик влажности почвы (или гигрометр) используется, как правило, для определения влажности почвы. Он пригодится для проектирования автоматической системы полива или отслеживания влаги в почве, где растут ваши растения.

На рисунке 1.3 представлен датчик влажности почвы *YL-69*, выбранный для использования в курсовом проекте.



Рисунок 1.3 – Датчик влажности почвы *YL-69*

Датчик влажности почвы состоит из двух частей: электронной платы (на рисунке 1.3 расположена справа) и измерительной части с двумя пластинами, которая определяет наличие воды (на рисунке 1.3 расположена слева). Чувствительность датчика влажности почвы настраивается встроенным потенциометром. Кроме того, датчик оснащен двумя светодиодами: первый загорается, когда на плату подается питание, а второй – при выводе цифровых данных [8].

Рисунок 1.4 отображает схему датчика *YL-69*.

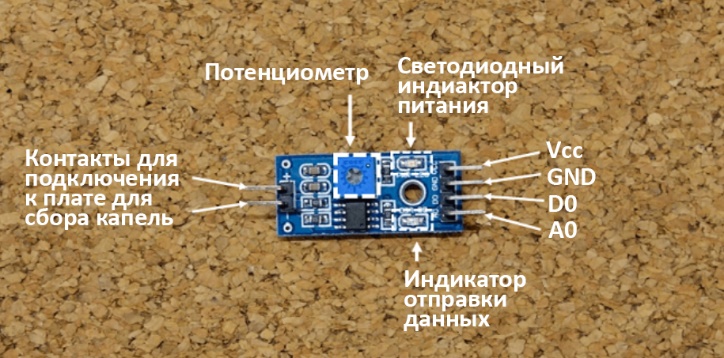


Рисунок 1.4 – Схема датчика влажности почвы *YL-69*

Датчик влажности почвы *YL-69* работает на основе измерения электрического сопротивления между двумя металлическими электродами, которые погружаются в почву. При наличии влаги вода проводит электрический ток, снижая сопротивление между электродами. В результате в влажной почве датчик выдает высокое значение напряжения на выходе, а в сухой – низкое. Это позволяет точно определить уровень влажности почвы и передать информацию в систему управления.

*YL-69* находит применение в различных областях, таких как автоматические системы полива, где он регулирует подачу воды в зависимости от влажности. В садоводстве и сельском хозяйстве датчик помогает мониторить условия роста растений и оптимизировать водоснабжение. Он также используется в гидропонных системах и научных исследованиях, где необходимо изучать влияние влажности на растения и экосистемы.

Выбор *YL-69* для проекта автоматической системы полива на базе *Arduino Nano* обусловлен несколькими факторами. Во-первых, датчик имеет простую конструкцию с тремя выводами (*VCC, GND, A0*), что облегчает его интеграцию с *Arduino*. Во-вторых, его аналоговый выход позволяет легко считывать значения с помощью встроенного АЦП *Arduino*, обеспечивая точные измерения. Третьим преимуществом является низкая стоимость, что делает его доступным для любительских и образовательных проектов. Кроме того, *YL-69* демонстрирует надежность и стабильные результаты при правильной эксплуатации, что делает его подходящим для длительного использования в автоматических системах полива. Наконец, датчик легко комбинируется с насосами и реле, необходимыми для автоматизации полива, что делает его идеальным выбором для данного проекта.

## 1.4 Насос для автоматической системы полива

Погружной водяной мини-насос на 5 В (*DC 5V*) представляет собой компактное устройство, предназначенное для перекачивания воды и других жидкостей. Он работает от постоянного тока 5 В, что делает его совместимым с *Arduino* и другими микроконтроллерами [9]. Этот насос отличается небольшими размерами, что позволяет легко интегрировать его в различные проекты, не занимая много места. Обычно такие насосы способны перекачивать от 100 до 200 литров в час, в зависимости от модели и условий работы, и могут поднимать воду на высоту до 1-2 метров, что делает их подходящими для большинства домашних и садовых приложений [10]. Изготавливается насос из высококачественных пластиковых материалов, устойчивых к коррозии, что увеличивает его долговечность. Благодаря прочности этой модели водяного мини-насоса можно обеспечить долговечную работу устройства для автоматического полива растений [11].

Изображение насоса представлено на рисунке 1.5.



Рисунок 1.5 – Погружной водяной мини-насос *DC5V, 120L-H*

В контексте автоматических систем полива этот насос идеально подходит для подачи воды к растениям, управляемых микроконтроллерами, такими как *Arduino*. Он будет выкачивать воду из емкости и через шланг для капельного полива будет орошать почву в горшке растения. Также он может использоваться в аквариумах для создания циркуляции воды, а также в моделях и макетах, где необходимо перекачивание жидкости. Простота подключения к *Arduino*, где насос можно напрямую подключить к 5 В и управлять с помощью цифровых выводов, делает его удобным для использования.

Выбор этого насоса для проекта автоматического полива на *Arduino Nano* обусловлен его надежностью, компактностью и эффективностью. Он обеспечивает достаточную производительность для полива растений и легко интегрируется в систему, что позволяет создать эффективное и автоматизированное решение для управления поливом.

## 1.5 Дисплей *LCD*1602 с *I*2*C* адаптером

*LCD*1602 с *I*2*C*-адаптером — это удобный модуль для отображения текстовой информации. Он состоит из *LCD*-дисплея с матрицей на 16 символов в 2 строки и *I*2*C*-адаптера, который упрощает подключение к микроконтроллерам. Основной управляющий элемент дисплея — контроллер *HD*44780, обеспечивающий работу матрицы. *I*2*C*-адаптер позволяет использовать интерфейс *I*2*C* вместо стандартного параллельного подключения, что делает модуль более компактным и простым в использовании [12].

Изображение дисплея с адаптером продемонстрировано на рисунке 1.6.

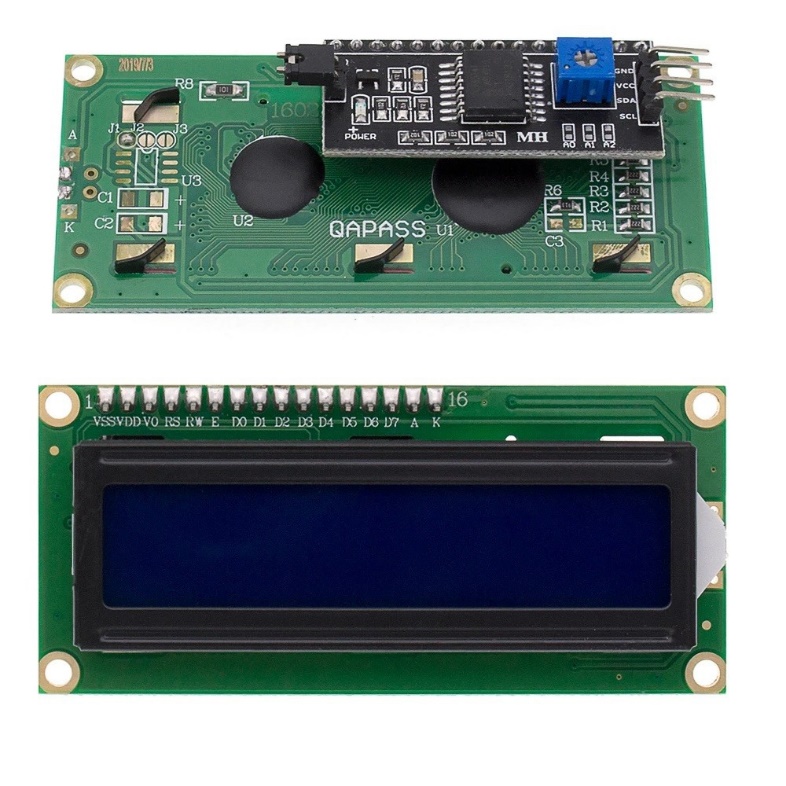


Рисунок 1.6 – Дисплей *LCD*1602 с *I*2*C*-адаптером

Выводы модуля с *I*2*C*:

– *VCC* – питание дисплея (5В).

– *GND* – земля.

– *SDA* – линия передачи данных по *I*2*C*.

– *SCL* – линия синхронизации по *I*2*C*.

Одним из ключевых преимуществ *LCD*1602 с *I*2*C* является значительная экономия пинов микроконтроллера: вместо 6–10 пинов для стандартного подключения требуется всего 2 (*SDA* и *SCL*). Это особенно полезно в проектах с ограниченным количеством выводов. Подключение дисплея максимально упрощено благодаря использованию готовых библиотек, таких как *LiquidCrystal\_I*2*C*.*h*, которые избавляют от необходимости сложной настройки. Также дисплей обладает регулируемой подсветкой и контрастностью, что обеспечивает комфортное восприятие информации. Его энергопотребление остается низким, что важно для автономных систем. *LCD*1602 идеально подходит для вывода числовых и текстовых данных, делая его универсальным решением для многих задач [13].

Использование этого дисплея в проекте оправдано его компактностью и универсальностью. Он позволяет эффективно отображать значения, полученные от датчиков, а также уведомления об ошибках. Это делает его не только элементом визуализации, но и важной частью интерфейса пользователя, обеспечивающей обратную связь. Удобство работы с данными датчиков помогает пользователю своевременно отслеживать состояние системы.

В данном проекте *LCD*1602 выполняет несколько задач. Он отображает показания влажности почвы, полученные от двух датчиков, а также выводит предупреждения о некорректной работе системы. Это позволяет пользователю в реальном времени контролировать процесс автоматического полива и при необходимости устранять возможные проблемы. Дисплей является важным элементом, обеспечивающим надежность и удобство эксплуатации всей системы.

## 1.6 Модуль реле

Реле – это электромеханическое устройство, предназначенное для управления высоковольтными или мощными нагрузками при помощи слабого электрического сигнала. Основными компонентами реле являются электромагнитная катушка, механический переключатель (контакты) и корпус, обеспечивающий защиту и изоляцию. При подаче напряжения на катушку происходит замыкание или размыкание контактов, что позволяет включать или отключать подключенные устройства, такие как насосы, лампы или двигатели. На рисунке 1.7 представлен модуль реле *JQC-3FF-S-Z 5VDC*, выбранный для использования в данном проекте.



Рисунок 1.7 – Модуль реле *JQC-3FF-S-Z 5VDC*

Основным преимуществом реле *JQC*-3*FF*-*S*-*Z* является его способность безопасно управлять цепями с высокими напряжением и током, используя при этом маломощный сигнал, поступающий от микроконтроллера. Реле поддерживает работу с напряжением катушки 5 В постоянного тока, что делает его совместимым с большинством популярных микроконтроллеров, таких как *Arduino*. Контакты реле рассчитаны на переключение токов до 10 А, что обеспечивает надежное управление мощными устройствами. Наличие гальванической развязки между управляющей и управляемой цепями увеличивает безопасность работы системы.

Реле *JQC*-3*FF*-*S*-*Z* 5*VDC* имеет входные и выходные контакты, которые разделены на две стороны: управляющую (входную) и коммутационную (выходную).

Входная сторона:

1 *VCC* (*Voltage* *Common* *Collector*):

Контакт для подключения питающего напряжения реле. Обычно это 5 В постоянного тока, что делает его совместимым с микроконтроллерами, такими как *Arduino*. Используется для питания электромагнитной катушки реле, которая обеспечивает переключение его контактов.

2 *GND* (*Ground*):

Контакт для подключения общего провода питания (земли). Обеспечивает замыкание электрической цепи питания катушки, необходимой для работы реле.

3 *IN* (*Input*):

Управляющий сигнал, подаваемый от микроконтроллера (например, через цифровой пин *Arduino*). Когда на этот пин подается высокий уровень напряжения (5 В), катушка реле активируется, замыкая или размыкая контакты на коммутационной стороне.

Выходная сторона:

1 NO (*Normally* *Open*):

Контакт, который в нормальном состоянии (при отсутствии напряжения на катушке) остается разомкнутым. Используется для подключения нагрузки, которая должна быть включена только при активации реле. Например, насос включится, когда реле получит управляющий сигнал.

2 *COM* (*Common*):

Общий контакт, который соединяется либо с *NO*, либо с *NC* в зависимости от состояния реле. К нему подключается общий провод нагрузки (например, один из контактов насоса).

3 *NC* (*Normally* *Closed*):

Контакт, который в нормальном состоянии (без подачи напряжения на катушку) остается замкнутым с общим контактом (*COM*). Используется для подключения нагрузки, которая должна быть включена, когда реле неактивно [14].

Использование реле *JQC*-3*FF*-*S*-*Z* в проекте объясняется его простотой и надежностью. Оно позволяет управлять включением и выключением погружного насоса для автоматического полива, защищая микроконтроллер от воздействия высокого тока или напряжения. Применение реле также делает систему более универсальной, так как оно может быть настроено для работы с различными типами нагрузок.

В данном проекте реле играет важную роль в управлении насосом. С его помощью микроконтроллер способен включать насос при недостаточной влажности почвы и отключать его при достижении необходимого уровня или по истечении установленного времени работы. Это обеспечивает автоматизацию системы полива, повышает ее эффективность и безопасность, а также уменьшает потребление энергии.

# 2 ПЛАТФОРМА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

В рамках данного курсового проекта была выбрана операционная система *Windows* 10. Эта ОС обладает рядом преимуществ, которые делают её идеальным выбором для разработки программного обеспечения для микроконтроллеров. *Windows* 10 обеспечивает стабильную и производительную среду, поддерживает широкий спектр программного обеспечения и драйверов, что упрощает взаимодействие с различными устройствами, включая *Arduino*.

## История операционной системы *Windows*

1. *Windows* 1.0 (Ноябрь 1985)

При первом релизе *Windows* операционная система была далека от современного понимания полноценной ОС. В то время она функционировала как «операционная среда» для *MS-DOS* и изначально была названа *Interface Manager*. Несмотря на свою простоту, первая версия уже включала ряд инновационных инструментов, таких как графический редактор *Windows Paint*, текстовый процессор *Windows Write* и известная настольная игра *Reversi*.

1. *Windows* *2.X* (Декабрь 1987)

Следующий крупный релиз *Windows* представил приложения *Excel* и *Word*, которые стали основными элементами в истории программного обеспечения. Не менее значимым в этом контексте является приложение *Aldus PageMaker*, ранее доступное только на платформе *Macintosh*, которое способствовало росту популярности *Windows* в 1987 году. Следует отметить, что, хотя *Aldus PageMaker* был доступен с версии 1.0, именно версия 2.0 обрела славу на платформе *Microsoft Windows*. Однако рост популярности *Windows* также стал причиной конфликта с *Apple*, которая заявила о нарушении патентов на многие элементы пользовательского интерфейса.

1. *Windows 3.X* (Май 1990)

Версия *Windows 3.X* обеспечила улучшение многозадачности, внедрение виртуальной памяти и обновление дизайна, что позволило интерфейсу *Windows* конкурировать с интерфейсом *Macintosh*. Вместе с *Windows* 3.1 было введено понятие «Мультимедийного ПК», где использование *CD-ROM* и звуковых карт стало основным трендом 1990-х годов. Продажа 10 миллионов копий версии 3.0 сделала её основным источником дохода для *Microsoft* и значительным успехом в мире информационных технологий.

1. *Windows NT* (Июль 1992)

В рамках сотрудничества с *IBM Microsoft* разработала преемника *DOS*, известного как *OS*/2, однако это сотрудничество было недолгим. Вместо этого была создана новая операционная система *Windows NT*, которая разрабатывалась параллельно с *Windows* 3.11 и впоследствии была объединена с ней в *Windows XP*. Благодаря улучшенной сетевой поддержке и новой файловой системе *NTFS*, *Microsoft* смогла стать основным игроком на серверном рынке.

1. *Windows* 95 (Август 1995)

*Windows* 95 реализовала идеи, представленные в версии *NT*, включая 32-разрядную архитектуру и улучшенное управление памятью. Тем не менее, необходимость поддержания обратной совместимости и не полностью адаптированный код привели к проблемам с производительностью и стабильностью. В более поздних обновлениях *Windows* 95 был интегрирован браузер *Internet Explorer* и улучшена поддержка *USB*.

1. *Windows* 98 (Июнь 1998)

С выходом *Windows* 98, разработанной под кодовым названием *Memphis*, *Microsoft* значительно улучшила поддержку *USB*, поскольку предыдущая версия не обеспечила её стабильной реализации. В этом выпуске также была представлена файловая система *FAT32*, что способствовало распространению дисковых разделов объемом более двух гигабайт. 1998 год ознаменовался юридической разборкой «Соединенные Штаты против *Microsoft*», касающейся легитимности предустановленного браузера *Internet Explorer*.

1. *Windows* 2000 (Февраль 2000)

В следующей версии *Windows NT* появилась новая служба – *Active Directory*. Несмотря на то что версия была нацелена на бизнес-рынок, *Windows* 2000 включала улучшенный *API DirectX*, что открыло новые возможности для запуска современных игр на компьютерах с *NT*. В то же время в этой версии была введена новая и спорная система активации продукта.

1. *Windows ME* (Сентябрь 2000)

*Microsoft* выпустила «*Millennium Edition*» (*ME*) как преемника *Windows* 98 и последнюю операционную систему на базе *DOS*. В данной версии акцент был сделан на мультимедийные возможности, включая *Windows Movie Maker* и обновленный медиаплеер *Windows Media Player* до версии 7. Кроме того, была представлена утилита для восстановления системы – *System Restore*.

1. *Windows XP* (Август 2001)

*Windows XP* объединила в себе лучшие черты как *Windows 95/98/ME*, так и *NT/2000*. Хотя на начальном этапе имелись проблемы безопасности, что привело к выходу нескольких пакетов обновлений, *Windows XP* все же стала флагманом среди операционных систем и сохраняла свою популярность в течение шести лет – дольше, чем любая другая версия *Microsoft Windows*.

1. *Windows Vista* (Январь 2007)

*Windows Vista* была представлена с совершенно новым дизайном, основанным на графическом интерфейсе *Windows Aero*. В этой версии было множество изменений, включая новую систему контроля учетных записей пользователей и улучшенные приложения, такие как *Windows Calendar* и *Windows DVD Maker*.

1. *Windows* 7 (Октябрь 2009)

*Windows* 7 явилась значительным улучшением во многих областях: она обеспечивала более быструю загрузку, поддержку *multi-touch* и улучшенное управление окнами. Однако новая система контроля учетных записей стала менее навязчивой, а боковая панель была полностью удалена.

1. *Windows* 8 (Октябрь 2012)

*Windows* 8 представила значительное визуальное обновление, включая новый интерфейс и поддержку полноэкранного режима. Также была внедрена поддержка *USB* 3.0 и запущен *Windows Store*.

1. *Windows* 10 (Июль 2015)

*Microsoft* приняла решение назвать последнюю версию «*Windows* 10», пропустив версию 9, что может быть связано с масштабом и важностью проекта. *Windows* 10 обеспечивает общую платформу для различных устройств, от смартфонов до персональных компьютеров. Среди новых функций можно выделить браузер *Edge*, поддержку входа с помощью отпечатков пальцев и распознавания лиц, а также умного личного помощника *Cortana*. Также важно отметить, что обновление с *Windows* 7 и 8 было бесплатным в течение первого года после релиза [15].

## Преимущества операционной системы *Windows*

Одним из основных преимуществ *Windows* 10 является её совместимость с *Arduino IDE*, которая активно используется для разработки программного обеспечения для микроконтроллеров *AVR*.

Кроме того, *Windows* 10 обеспечивает хорошую поддержку периферийных устройств, таких как *USB*-устройства, что важно для подключения микроконтроллеров и других компонентов системы. Наличие регулярных обновлений и улучшений безопасности также делает эту ОС надежной для разработки. Таким образом, выбор *Windows* 10 для реализации проекта обусловлен её стабильностью, совместимостью с необходимым программным обеспечением и удобством работы для разработчиков.

## *Arduino* *IDE*

Как было сказано ранее, для разработки программного обеспечения для микроконтроллеров AVR часто используется Arduino IDE. Эта платформа предлагает удобный интерфейс для написания, компиляции и загрузки кода на микроконтроллеры. Arduino IDE поддерживает множество библиотек и примеров, что облегчает процесс разработки и позволяет быстро реализовывать идеи. Активное сообщество разработчиков обеспечивает доступ к обширной базе знаний и решениям для различных задач, поэтому данная платформа была выбрана для разработки проекта.

Кроме того, Arduino IDE имеет интуитивно понятный интерфейс, что делает его доступным даже для начинающих разработчиков. Возможность легко интегрировать различные модули и сенсоры позволяет создавать сложные проекты с минимальными усилиями, что значительно ускоряет процесс прототипирования и тестирования.

Интерфейс *IDE* изображен на рисунке 2.1.

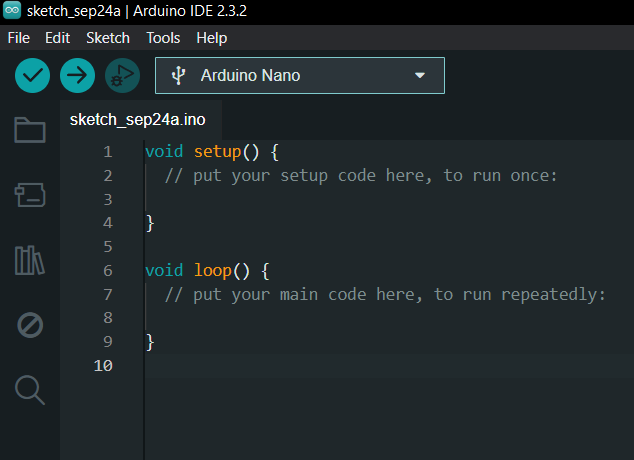


Рисунок 2.1 – Интерфейс *Arduino IDE*

Среди ключевых возможностей *Arduino IDE* – большое количество встроенных библиотек, что упрощает работу с различными датчиками, модулями и компонентами. Большое сообщество пользователей гарантирует обширные учебные материалы, форумы и примеры кода, что значительно облегчает процесс обучения и решения возникающих проблем. Также в среде разработки есть инструменты для мониторинга последовательного порта, что помогает в отладке кода и отображении данных с датчиков, что делает процесс разработки более удобным.

*Arduino IDE* предлагает множество функций, таких как редактирование кода с подсветкой синтаксиса и автоматическим завершением, что упрощает написание программ. Пользователи могут создавать и управлять несколькими проектами, включая импорт и экспорт библиотек, что значительно упрощает организацию работы. Язык программирования, основанный на *C*/*C+*+, легко осваивается и позволяет работать с микроконтроллерами. Среда также предоставляет возможность настраивать параметры компиляции и загрузки в зависимости от конкретных нужд проекта [16].

Дополнительно *Arduino IDE* поддерживает сторонние библиотеки и расширения, что позволяет пользователям улучшать функциональность среды. Можно использовать графические библиотеки для создания интерфейсов и отображения данных, а также есть возможность эмуляции проектов, что позволяет тестировать код без необходимости загружать его на физическую плату [17].

В итоге, *Arduino IDE* является мощным инструментом для разработки проектов на базе *Arduino*, благодаря своей простоте, множеству возможностей и поддержке сообщества, что делает её идеальной как для начинающих, так и для опытных разработчиков, стремящихся создавать разнообразные электронные проекты.

1. **ТЕОРЕТИЧЕСКОЕ ОБОСНОВАНИЕ РАЗРАБОТКИ ПРОГРАММНОГО ПРОДУКТА**
   1. **Обоснование необходимости разработки**

Автоматическая система полива комнатных растений на платформе Arduino Nano является современным и удобным решением, которое отвечает на ключевые вызовы, с которыми сталкиваются владельцы домашних растений в условиях ускоряющейся урбанизации. В эпоху, когда время становится одним из самых ценных ресурсов, а популярность домашнего садоводства стремительно растет, возникает необходимость в автоматизации процессов ухода за растениями. Особенно актуальной эта проблема становится для людей с плотным графиком работы, которые не всегда могут уделить должное внимание своим растениям, а также для тех, кто часто отсутствует дома в связи с командировками, поездками или другими обстоятельствами. Без должного ухода растения рискуют страдать от нехватки влаги или, напротив, от избыточного полива, что негативно сказывается на их состоянии и может привести к гибели.

Создание системы автоматического полива на платформе Arduino Nano позволяет не только решить проблему ухода за растениями в отсутствие владельца, но и оптимизировать процесс полива в целом. Благодаря применению современных технологий, система может обеспечивать своевременное и точное увлажнение почвы в зависимости от текущего состояния растения, его вида и степени влажности грунта. Это значительно снижает риски, связанные как с пересушиванием, так и с переувлажнением почвы. Недостаток влаги может замедлить рост растения, привести к увяданию листьев и снижению его жизнеспособности, тогда как избыток воды способен вызвать гниение корневой системы и развитие грибковых заболеваний. Автоматизация процесса позволяет избежать этих проблем, создавая комфортные условия для растений, что способствует их здоровью, пышному росту и долгому сроку жизни.

Одним из ключевых преимуществ системы автоматического полива является возможность получения данных о влажности почвы в режиме реального времени. Благодаря использованию датчиков влажности, система может оперативно реагировать на изменения уровня влаги в почве, обеспечивая растения необходимым количеством воды именно тогда, когда это действительно необходимо. Такой подход не только улучшает состояние растений, но и способствует формированию у пользователей более ответственного и внимательного отношения к уходу за зелеными насаждениями. Домашние растения являются неотъемлемой частью уюта и благоприятного микроклимата, а их здоровье напрямую зависит от качества ухода. Система автоматического полива позволяет минимизировать человеческий фактор и обеспечить стабильный уход, даже если владелец растений не имеет достаточного опыта или знаний в области садоводства.

В качестве метода орошения для системы был выбран прикорневой капельный полив. Этот метод считается одним из наиболее эффективных и экологически рациональных способов орошения. Прикорневой капельный полив обеспечивает точную подачу воды непосредственно в зону корней растений, что позволяет значительно снизить потери воды из-за испарения или стока. Это особенно важно в условиях ограниченного водного ресурса, когда необходимо использовать воду максимально рационально. Точная подача воды способствует равномерному увлажнению почвы, улучшая доступ корневой системы растений как к влаге, так и к питательным веществам, что положительно сказывается на их росте и состоянии. При этом метод предотвращает переувлажнение грунта, которое часто становится причиной различных заболеваний растений, в частности, грибковых инфекций.

Общая схема прикорневого капельного полива продемонстрирована на рисунке 3.1.

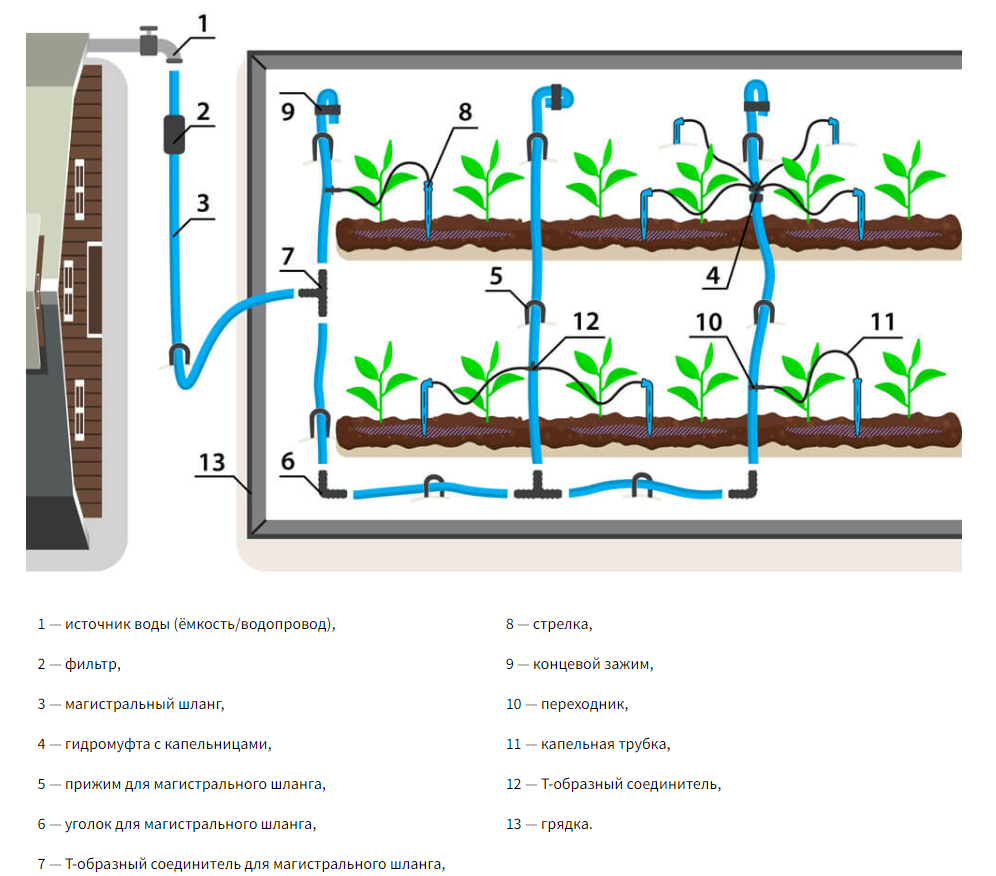


Рисунок 3.1 – Обобщенная схема автоматического прикорневого полива

При подобном методе полива растений вода автоматически поступает напрямую к корням, минимизируя участие человека в данном процессе [18].

* 1. **Технологии программирования, используемые для решения поставленных задач**

Разработка программного обеспечения для системы полива осуществляется с использованием *Arduino IDE*, что предоставляет разработчику широкие возможности для реализации функционала. Использование данного инструмента позволяет легко интегрировать различные сенсоры и модули, что критически важно для создания многофункциональной системы. Язык программирования *C++* обеспечивает высокую производительность и эффективность работы алгоритмов, что позволяет обрабатывать данные от датчиков и управлять исполнительными устройствами.

*C++* ­– это компилируемый язык программирования общего назначения, который сочетает свойства высокоуровневых и низкоуровневых языков программирования. В сравнении с его предшественником, языком *C*, он уделяет больше внимания поддержке объектно-ориентированного и обобщённого программирования. Название языка происходит от *C*, где унарный оператор ++ обозначает инкремент переменной. Благодаря своей универсальности, *C++* нашел широкое применение в разработке программного обеспечения, включая прикладные программы, операционные системы, драйверы устройств, игры и системы автоматизации.

Особую популярность *C++* получил при работе с платформами вроде *Arduino*. Его высокоэффективный синтаксис, базирующийся на языке *C*, позволяет создавать производительный код с низким уровнем потребления ресурсов, что крайне важно для микроконтроллеров. Объектно-ориентированная структура языка упрощает работу с периферийными устройствами через классы и библиотеки, а строгая типизация помогает минимизировать ошибки в коде. Это делает *C++* идеальным выбором для разработки надежных и масштабируемых проектов на *Arduino*.

*C++* был создан в начале 1980-х годов сотрудником *Bell Laboratories* Бьёрном Страуструпом. Изначально язык задумывался как расширение *C* для решения собственных задач разработчика, и первые версии, известные как «*C* с классами», начали появляться с 1980 года. Базовый язык *C*, широко применявшийся в системе *UNIX*, привлекал своей скоростью, многофункциональностью и переносимостью. Добавив к нему возможности работы с объектами и классами, Страуструп заложил основу нового языка программирования, который сохранил совместимость с *C*. В 1983 году язык был переименован в *C++*, и в него добавили такие новшества, как виртуальные функции, перегрузка операторов, ссылки и многое другое. Первый коммерческий выпуск *C++* состоялся в октябре 1985 года, положив начало его широкому распространению.

Различные реализации *C++* сегодня доступны как в бесплатных, так и в коммерческих версиях. Среди них проекты *GNU*, *Microsoft* и *Embarcadero*, а проект *GNU* активно поддерживает свободное программное обеспечение. Широкий выбор инструментов и высокая производительность делают *C++* востребованным как в крупных разработках, так и в проектах на микроконтроллерах, таких как *Arduino* [19].

Для реализации алгоритмов контроля влажности будут использоваться как аналоговые, так и цифровые датчики, что позволит достигать высокой точности в измерениях и реакциях системы на изменения в состоянии почвы. В дополнение к этому, возможно использование библиотек для работы с *LCD*-дисплеями, что создаст удобный интерфейс для визуализации данных и мониторинга состояния системы в реальном времени. Таким образом, выбранные технологии программирования не только позволяют эффективно решать поставленные задачи, но и открывают возможности для дальнейшего расширения функционала системы.

* 1. **Связь архитектуры вычислительной системы с разрабатываемым прибором**

Архитектура вычислительной системы играет ключевую роль в работе автоматической системы полива, реализованной на базе микроконтроллера *Arduino* *Nano*. Этот микроконтроллер использует архитектуру *Harvard*, которая обеспечивает разделение памяти данных и программ, что повышает производительность устройства и позволяет эффективно реализовывать алгоритмы управления.

Микроконтроллер *ATmega*328*P*, на котором основана *Arduino* *Nano*, обладает ограниченными ресурсами: 32 КБ флеш-памяти и 2 КБ оперативной памяти. Эти характеристики требуют оптимизации программного обеспечения для обеспечения компактности кода и эффективного использования доступной памяти. В системе полива программное обеспечение отвечает за сбор данных с датчика влажности, обработку этих данных, управление насосом через реле, а также за обработку ошибок и выдачу информации на *LCD*-дисплей. Использование эффективных алгоритмов и библиотек позволяет реализовать все функции в условиях ограниченных вычислительных возможностей.

Программное обеспечение активно использует *GPIO*-пины микроконтроллера для взаимодействия с периферийными устройствами. Например, данные с датчиков влажности поступают через аналоговые входы, а управление насосом осуществляется через цифровой выход, связанный с реле. Кроме того, связь с *LCD*-дисплеем с интерфейсом *I*2*C* минимизирует количество используемых пинов, что особенно важно в компактных устройствах. Такие особенности архитектуры *Arduino* *Nano* позволяют интегрировать все необходимые компоненты в одном устройстве.

Прерывания и таймеры микроконтроллера играют важную роль в работе системы полива. Таймеры используются для регулирования времени работы насоса и времени отката, в течение которого устройство игнорирует показания датчиков, чтобы предотвратить частые включения и выключения, ведущие к износу насоса. Прерывания обеспечивают реакцию на критические события, такие как резкие изменения во входных данных, что позволяет системе своевременно реагировать на изменения условий.

Процессор *Arduino* *Nano* работает на частоте 16 МГц, что накладывает ограничения на сложность используемых алгоритмов. Для системы полива применяются простые, но эффективные методы обработки данных, которые позволяют минимизировать вычислительные затраты. Например, данные с датчиков влажности фильтруются для исключения аномальных значений, что повышает точность работы системы. Эти алгоритмы разработаны с учетом ограничений архитектуры, что позволяет достичь надежной и стабильной работы.

Таким образом, архитектура микроконтроллера *Arduino* *Nano* не только определяет аппаратные возможности системы автоматического полива, но и формирует подход к разработке программного обеспечения. Ограниченные ресурсы стимулируют использование оптимизированных алгоритмов и эффективное управление периферийными устройствами, что в совокупности обеспечивает точность, надежность и энергосбережение системы.

* 1. **Актуальность разработки системы**

Разработка системы автоматического полива является актуальной задачей в свете современных тенденций к автоматизации бытовых процессов и рациональному использованию природных ресурсов. Система предназначена для облегчения ухода за комнатными растениями, что особенно важно для людей с напряжённым графиком, ограниченным временем или недостаточными знаниями в области садоводства. Она позволяет автоматически обеспечивать растения необходимым уровнем влажности почвы, избавляя пользователя от необходимости регулярного контроля.

На фоне растущего интереса к экологически чистым технологиям и устойчивому ведению хозяйства, автоматические системы полива становятся востребованными инструментами. Они обеспечивают не только удобство и комфорт, но и рациональное использование воды, минимизируя её перерасход. Это особенно важно в условиях глобального дефицита водных ресурсов и изменений климата, когда каждый вклад в экономию природных богатств приобретает значимость. Система автоматического полива вносит вклад в экологичность современного садоводства, поддерживая оптимальные условия для роста растений без избыточных затрат воды.

Для городских жителей такая система становится инновационным решением, которое упрощает уход за зелёными насаждениями в домашних условиях. Люди, живущие в условиях мегаполисов, часто сталкиваются с нехваткой времени на регулярный уход за растениями, а также с трудностями в поддержании благоприятного микроклимата для них. Система автоматического полива обеспечивает стабильное увлажнение почвы даже в периоды отсутствия владельцев, что делает её незаменимой для использования в квартирах, офисах и других помещениях.

Кроме того, современное программное обеспечение и аппаратная часть системы предоставляют пользователям гибкость в настройках. Это позволяет индивидуально адаптировать параметры полива под конкретные виды растений и их потребности. Возможность точного контроля влажности почвы и автоматическое прекращение полива при достижении заданных значений не только упрощают уход за растениями, но и продлевают срок службы используемых компонентов, таких как насосы и датчики [20].

Таким образом, разработанная система автоматического полива комнатных растений отвечает современным требованиям к автоматизации, удобству и экологичности. Она станет полезным инструментом как для любителей растений, так и для профессиональных садоводов, предоставляя возможности для более эффективного управления водными и временными ресурсами. В результате пользователи смогут создавать благоприятные условия для роста растений, сократив при этом свои трудозатраты и улучшив общий комфорт ухода. Сочетание инновационных технологий и практических решений делает эту разработку актуальной, востребованной и перспективной в современном мире.

1. **ПРОЕКТИРОВАНИЕ ФУНКЦИОНАЛЬНЫХ ВОЗМОЖНОСТЕЙ ПРОГРАММЫ**

Программное обеспечение автоматической системы полива комнатных растений разработано для обеспечения автономного ухода за растениями путем анализа текущих показателей влажности почвы и принятия решений по поливу. Основные функции ПО ориентированы на мониторинг состояния почвы, управление поливом, корректировку частоты замеров, а также диагностику ошибок, связанных с показаниями датчиков.

1. Мониторинг состояния почвы

Главная функция программного обеспечения заключается в непрерывном мониторинге показателей влажности почвы с использованием аналоговых датчиков. Программа выполняет регулярные замеры с подключенных датчиков, каждый из которых передает аналоговые сигналы, отражающие текущий уровень влажности почвы. Сигналы с датчиков поступают на аналоговые входы микроконтроллера и преобразуются в цифровое значение, которое затем сравнивается с заранее установленным порогом влажности. Использование аналоговых датчиков позволяет задавать гибкие пороговые значения влажности и легко адаптировать систему под различные типы растений с разными потребностями к количеству воды в почве.

Программа считывает значения с интервалом, установленным в коде, и анализирует их на предмет отклонений от нормы. Это позволяет системе эффективно контролировать состояние почвы и подстраиваться под изменяющиеся условия, включая температуру и влажность воздуха, которые также влияют на скорость высыхания почвы.

1. Автоматизированное управление поливом

Важной функцией системы является автоматическое управление насосом для полива растений в зависимости от показаний датчиков влажности. Когда уровень влажности почвы опускается ниже установленного минимального значения, программа запускает насос на фиксированный промежуток времени, обеспечивая подачу воды в почву. После завершения работы насоса наступает время отката, в течение которого насос игнорирует показания датчиков. Это предотвращает частое включение и выключение устройства, уменьшая его износ и продлевая срок службы. По истечении времени отката насос может быть повторно включен, если показатели датчиков по-прежнему указывают на необходимость полива. Такой подход помогает оптимизировать энергопотребление системы.

1. Настраиваемая частота замеров

Программное обеспечение включает функцию регулировки частоты замеров данных с датчиков. Настраиваемая частота позволяет адаптировать систему для разных типов растений, так как каждому виду требуется индивидуальный подход к поливу. Частота замеров устанавливается непосредственно в коде и может быть изменена на более высокую или низкую в зависимости от потребностей растения и окружающих условий.

Такой подход оптимизирует полив в зависимости от влажности воздуха и температуры, которые влияют на высыхание почвы. Для повышения точности замеров система делает паузу между циклами опроса датчиков, что позволяет достигать необходимой актуальности данных и при этом снижать нагрузку на микроконтроллер.

1. Аварийная индикация и обработка ошибок

Для обеспечения надежной и бесперебойной работы предусмотрены алгоритмы обработки ошибок и диагностики показателей датчиков. Например, в случае значительного отклонения показаний от типичных значений программа может интерпретировать это как сигнал о неисправности датчика. В коде предусмотрены функции, которые проверяют аномальные значения влажности, и при их обнаружении программа прекращает подачу воды, избегая ненужного расхода ресурса и возможного переувлажнения почвы. Эти меры помогают предотвратить нежелательные воздействия на растения в случае сбоев в системе или выхода датчиков из строя.

Данная функция, реализованная в программе, является дополнительным уровнем защиты для растений, так как позволяет избежать непредвиденных ситуаций и обеспечивает надежность работы системы даже при возникновении аппаратных неисправностей.

Таким образом, программное обеспечение автоматической системы полива растений позволяет организовать удобный и точный уход за растениями, сводя участие человека к минимуму.

1. **АРХИТЕКТУРА РАЗРАБАТЫВАЕМОЙ ПРОГРАММЫ**

* 1. **Общая структура системы автоматического полива**

Система автоматического полива представляет собой компактное устройство, которое предназначено для автономного управления процессом увлажнения почвы. Устройство выполнено в виде единого модуля, состоящего из центрального микроконтроллера (*Arduino* *Nano*), датчиков влажности почвы, насоса для подачи воды, реле для управления насосом, а также *LCD*-дисплея с интерфейсом *I*2*C* для отображения текущего состояния системы и сообщений об ошибках. Общая схема подключения всех элементов системы автоматического полива представлена на рисунке 5.1.



Рисунок 5.1 – Система автоматического полива изнутри

Визуально система представляет собой небольшой пластиковый корпус, в котором размещены основные элементы устройства. Снаружи к корпусу подключаются насос, питающий воду к растениям, датчики влажности, размещенные в почве, и источники питания. Дисплей установлен на верхней панели корпуса, что позволяет легко отслеживать текущие показатели. Дополнительно в конструкции предусмотрен переключатель, позволяющий независимо от программного кода отключить насос и включить его обратно при необходимости.

На рисунке 5.2 изображена конечная версия системы автоматического полива.



Рисунок 5.2 – Система автоматического полива в сборке

Система включает в себя электронную часть, обеспечивающую управление, и механическую часть, отвечающую за доставку воды. Такой подход позволяет организовать автоматический полив с минимальным участием пользователя, увеличивая удобство и эффективность ухода за растениями.

Датчики влажности почвы, используемые в проекте, позволяют в реальном времени отслеживать уровень влаги в почве и на основе полученных данных принимать решения о необходимости полива. Схема подключения датчиков к микроконтроллеру представлена на рисунке 5.3.

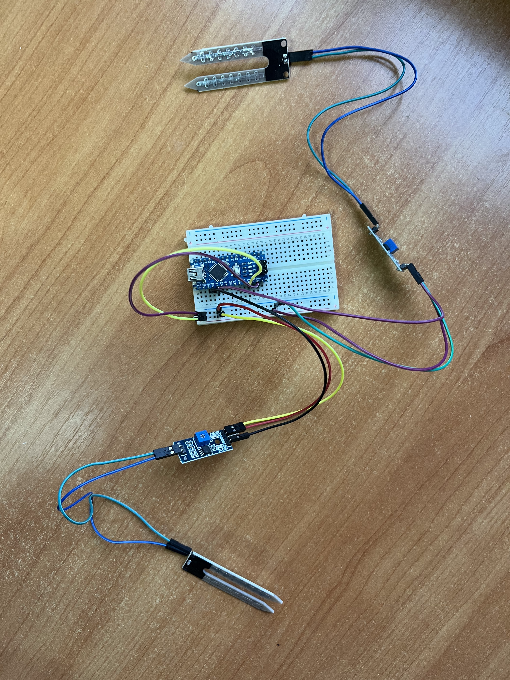


Рисунок 5.3 – Подключение датчиков к микроконтроллеру

Исполнительные механизмы, такие как насос и *LCD-*дисплей, будут выполнять различные действия в зависимости от данных, получаемых от датчиков. Схема подключения дисплея к микроконтроллеру *Arduino* представлена на рисунке 5.4.

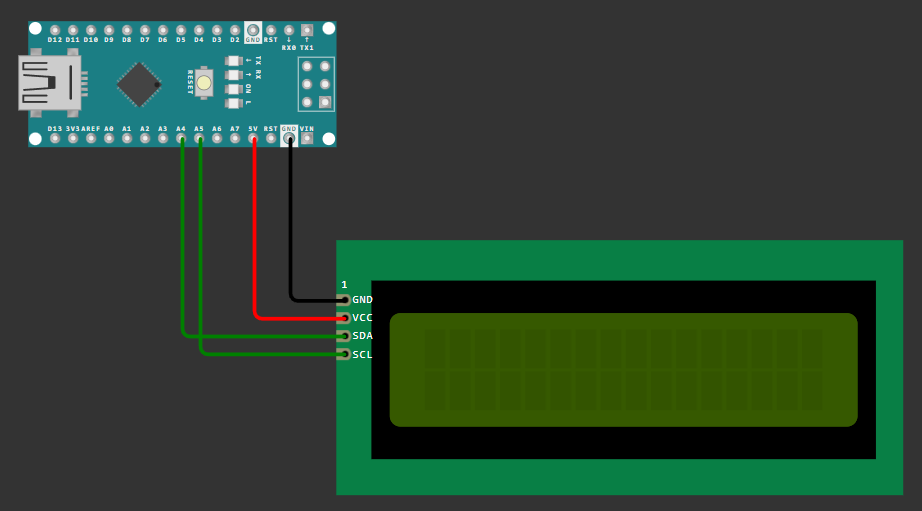


Рисунок 5.4 – Схема подключения дисплея к контроллеру

Для подключения *LCD* экрана с *I*2*C* адаптером к микроконтроллеру *Arduino* *Nano* необходимо корректно соединить пины экрана с соответствующими пинами на плате *Arduino*.

1 *GND* (Земля): Этот контакт экрана следует подключить к контакту *GND* на *Arduino* *Nano*. Подключение землей необходимо для обеспечения общего заземления всей электрической схемы, что важно для стабильной работы устройства.

2 *VCC* (Питание): Пин *VCC* на экране подает напряжение 5 В для работы дисплея. Его следует соединить с контактным пином 5*V* на плате *Arduino* *Nano*. Это подключение обеспечит необходимое питание для экрана.

3 *SDA* (*Data* *Line*): Контакт *SDA* используется для передачи данных между дисплеем и микроконтроллером. Для *Arduino* *Nano* этот пин следует подключить к пину *A*4, который *Arduino* *Nano* использует для работы с *I*2*C* и который является пином для линии данных в интерфейсе *I*2*C*.

4 *SCL* (*Clock* *Line*): Пин *SCL* выполняет функцию синхронизации передачи данных и подключается к пину *A*5 на *Arduino* *Nano*. Этот пин служит для передачи тактовых импульсов, необходимыми для правильной синхронизации обмена данными между микроконтроллером и дисплеем.

Для подключения насоса через реле к микроконтроллеру *Arduino* необходимо правильно соединить пины реле с соответствующими пинами на плате *Arduino*, а также учесть подключение насоса к контактам реле.

Схема данного подключения продемонстрирована на рисунке 5.5.

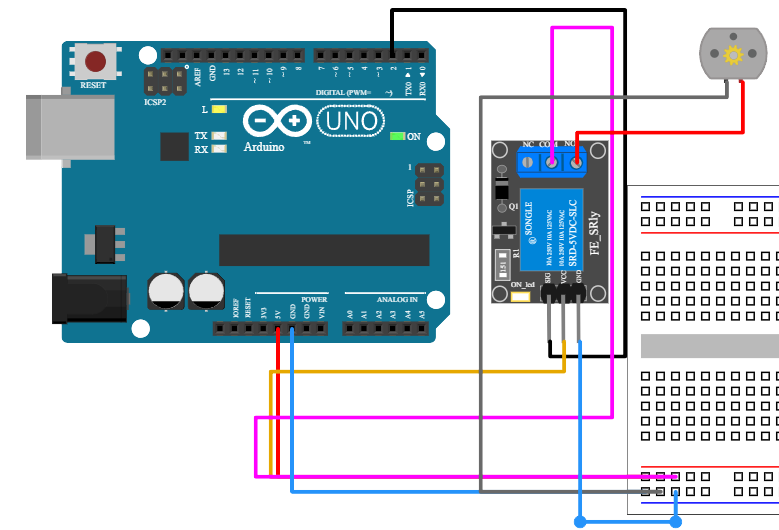


Рисунок 5.5 – Схема подключения насоса к контроллеру

1 *GND* (Земля): Контакт *GND* на реле должен быть подключен к пину *GND* на *Arduino* *Nano*. Это подключение обеспечивает общий уровень заземления, что необходимо для стабильной работы всей схемы.

2 *VCC* (Питание катушки реле): Пин *VCC* на реле должен быть соединен с пином 5*V* на плате *Arduino* *Nano*. Это питание необходимо для активации катушки реле, которая будет управлять переключением состояния контактов реле.

3 *IN* (Управляющий сигнал): Контакт *IN* на реле отвечает за получение сигнала от *Arduino*, который активирует реле и замыкает/размыкает его контакты. Для подключения этого пина следует использовать один из цифровых пинов на *Arduino* *Nano*, например, пин *D*2. Этот пин будет использоваться для подачи сигнала на реле с помощью простого *HIGH*/*LOW* сигнала, который будет управлять включением/выключением насоса.

4 *COM* (Общий контакт): Контакт *COM* на реле используется для подключения одной из сторон насоса. Этот пин необходимо соединить с положительным проводом питания насоса.

5 *NO* (Нормально открытый контакт): Контакт *NO* используется для подключения второго провода питания насоса. Он замкнется на пин *COM*, когда реле активируется, и разомкнется при его деактивации. Поэтому пин *NO* следует соединить с положительным проводом питания, который ведет к насосу.

6 *NC* (Нормально закрытый контакт): Контакт *NC* не используется в данной схеме, так как насос должен работать только при активации реле. Этот пин не подключается, поскольку в данном проекте не требуется замыкание цепи в неактивном состоянии реле.

Архитектура программного обеспечения автоматической системы полива комнатных растений включает несколько функциональных блоков, каждый из которых отвечает за определенные задачи, такие как мониторинг датчиков, анализ данных, управление поливом и обработка ошибок. Эта структура обеспечивает модульность и гибкость кода, что позволяет легко вносить изменения и оптимизации в зависимости от требований системы.

Программа состоит из следующих основных модулей:

1. Инициализация системы: на этапе запуска программы происходит настройка микроконтроллера, установка начальных параметров работы датчиков, а также заданы пороговые значения влажности для автоматического управления поливом.
2. Модуль замера влажности: основной блок, ответственный за считывание данных с датчиков влажности, подключенных к аналоговым входам. Он преобразует аналоговые сигналы в цифровое значение и передает их для дальнейшего анализа.
3. Анализ данных и принятие решений: этот модуль обрабатывает данные с датчиков, сравнивая их с установленным порогом. Если уровень влажности почвы ниже заданного значения, модуль отправляет сигнал на включение насоса для полива.
4. Управление насосом: модуль управления насосом выполняет команды, поступающие от блока принятия решений. В зависимости от результата анализа он включает или выключает насос.
5. Обработка ошибок: в этом модуле происходит проверка показателей на возможные неисправности. Например, если показания датчиков выходят за пределы допустимого диапазона, программа фиксирует это как ошибку, отключая насос для предотвращения повреждений.

Эта модульная структура упрощает разработку, отладку и масштабирование программы, так как каждый компонент можно легко модернизировать или заменять без нарушения функциональности всей системы.

* 1. **Описание функциональной схемы программы**

Функциональная схема алгоритма автоматической системы полива для комнатных растений с использованием Arduino Nano отражает основные этапы выполнения программы и взаимодействия между блоками системы. Функциональная схема алгоритма представлена в приложении В.

Схема включает следующие ключевые блоки:

1. Запуск программы и инициализация значений

Первый блок алгоритма отвечает за подготовку системы автоматического полива к работе. Его цель – настроить все аппаратные и программные компоненты таким образом, чтобы система могла функционировать автономно, стабильно и корректно выполнять свои задачи.

На этапе запуска система активируется при подаче питания от блока питания. В первую очередь производится инициализация управляющих компонентов, включая насос и датчики влажности почвы. Насос определяется как устройство, которое будет включаться и выключаться в зависимости от управляющих сигналов, поэтому ему задаётся начальное состояние – насос остаётся выключенным до момента анализа показаний с датчиков.

Датчики влажности подготавливаются для работы. Они будут считывать данные о влажности почвы в зоне своей установки. Эти данные интерпретируются в виде числовых значений, которые станут основой для принятия решений в алгоритме полива. Важно, чтобы датчики были готовы к корректному измерению с учётом возможных отклонений или погрешностей.

Для взаимодействия с пользователем используется дисплей, который инициируется на этапе запуска. Он предназначен для отображения информации о текущем состоянии системы: показаний датчиков, состояния насоса, а также предупреждений в случае выявления неисправностей. Отображение данных позволяет пользователю в любой момент узнать, как работает система, без необходимости подключения к компьютеру.

Кроме того, программа устанавливает начальные значения для всех внутренних переменных и параметров, которые будут использоваться в алгоритме. Это необходимо для обеспечения корректной работы системы, предотвращения сбоев и подготовки к дальнейшему анализу данных и управлению процессом полива.

На данном этапе завершается базовая настройка системы, и управление передаётся следующему блоку алгоритма, отвечающему за анализ показаний с датчиков.

1. Анализ значений с датчиков

Второй блок алгоритма является центральным элементом системы автоматического полива, так как именно на этом этапе происходит обработка данных с датчиков влажности почвы. Этот этап направлен на определение текущего состояния грунта и выявление необходимости включения насоса.

После инициализации системы показания с подключённых датчиков начинают считываться с заданной периодичностью. Каждый датчик измеряет влажность почвы в своей зоне покрытия, передавая значения в цифровом виде. Эти значения сравниваются с установленными порогами, которые определяют допустимые границы влажности. Например, если влажность ниже определённого порога, это сигнализирует о необходимости полива.

На этапе анализа система также проверяет корректность показаний с каждого датчика. Значения выходят за допустимые границы (например, аномально низкие или высокие) в случае неисправности или некорректной работы датчика. Если такое отклонение обнаруживается, программа определяет датчик как невалидный и исключает его данные из расчётов. Это позволяет системе продолжить работу, опираясь на показания только исправных датчиков, что повышает её устойчивость к сбоям.

Для дальнейшего управления насосом производится интегральная оценка состояния почвы. Если оба датчика работают корректно, система вычисляет усреднённое значение влажности. В случае, если один из датчиков выходит из строя, решение принимается на основе данных с оставшегося исправного датчика. При обнаружении неисправности сразу обоих датчиков система сигнализирует об ошибке, выводя предупреждение на дисплей, и останавливает насос до устранения проблемы.

1. Синхронизация работы насоса и экрана

Третий блок алгоритма отвечает за согласованное управление работой дисплея и насоса, что позволяет обеспечить удобство контроля и эффективность работы системы. Синхронизация необходима для предотвращения одновременного использования ресурсов дисплея и насоса, а также для минимизации энергопотребления.

Во время работы насоса дисплей отключает свою подсветку и перестаёт выводить информацию, чтобы уменьшить нагрузку на источник питания. Это особенно важно в условиях автономной работы от блока питания, где каждый элемент системы должен использовать ресурсы максимально эффективно. Выключение дисплея также снижает визуальный шум, так как пользователь вряд ли будет обращаться к информации на экране, пока насос работает.

Когда насос отключается, дисплей возвращается в активное состояние, включая подсветку и обновляя информацию о текущем состоянии системы. На экране вновь отображаются показания датчиков влажности и возможные предупреждения о неисправностях, если они были обнаружены в предыдущих блоках. Этот процесс происходит автоматически, без вмешательства пользователя.

Кроме того, синхронизация дисплея и насоса включает временные задержки и учёт состояний, чтобы переключение между режимами происходило плавно и без сбоев. Например, после завершения полива насос может быть заблокирован на определённое время, чтобы предотвратить его повторное включение до следующего цикла анализа. В этот период дисплей остаётся активным, предоставляя информацию о текущем статусе блокировки.

1. Проверка значений с датчиков

Четвёртый блок алгоритма занимается проверкой данных, поступающих с датчиков влажности почвы, с целью гарантии корректности работы системы и исключения ошибок, связанных с неисправностями или аномальными показаниями. Этот этап играет ключевую роль в обеспечении надёжности системы автоматического полива, так как от качества проверки зависит точность принятия решений.

Сначала каждое измерение, полученное от датчиков, сравнивается с заранее установленными допустимыми пределами. Эти пределы определяют минимальное и максимальное значения, которые могут быть физически возможны для корректно работающих датчиков. Например, если значение влажности выходит за границы (аномально низкое или высокое), это сигнализирует о возможной неисправности датчика или ошибке в измерении.

Если один из датчиков обнаруживает аномалию, система автоматически помечает его как невалидный и исключает его данные из дальнейшего анализа. Такое исключение предотвращает ситуацию, когда ошибочные показания могли бы повлиять на управление насосом, приводя к избыточному или недостаточному поливу. Однако если оба датчика оказываются невалидными, система сигнализирует о критической ошибке, отображая сообщение об этом на экране, и останавливает насос до устранения проблемы.

Кроме того, проверка значений включает диагностику ситуации, когда оба датчика работают корректно, но их показания значительно различаются. Это может быть связано с неравномерным распределением влаги в почве или локальными особенностями датчиков. В таком случае система может принимать решение на основе усреднённого значения или, при слишком большой разнице, сигнализировать о возможной проблеме.

Результаты проверки отображаются на экране в реальном времени, что позволяет пользователю оперативно отслеживать состояние системы. Если ошибки отсутствуют, выводится сообщение о стабильной работе. Если же обнаружены неисправности, на дисплее указывается, какой именно датчик вызывает проблему, или отображается общее предупреждение при неисправности обоих.

1. Управление работой насоса

Пятый блок алгоритма отвечает за управление насосом, который осуществляет подачу воды для автоматического полива. Этот блок выполняет ключевую функцию в системе, обеспечивая запуск и остановку насоса в зависимости от состояния почвы, определяемого по данным, поступающим с датчиков влажности.

После анализа данных, выполненного предыдущими блоками, система принимает решение о необходимости полива. Если показания датчиков указывают на недостаточную влажность почвы (значения превышают заданный порог), система проверяет текущее состояние насоса. Если насос выключен, а также отсутствует режим блокировки, он включается. Включение сопровождается регистрацией времени начала работы, что позволяет системе контролировать продолжительность полива. Это гарантирует, что насос не будет работать дольше предписанного времени, предотвращая переувлажнение почвы.

Во время работы насоса отключается подсветка дисплея, чтобы снизить энергопотребление, так как система предполагает автономное питание от блока питания. Однако насос продолжает функционировать в соответствии с заранее установленным таймером, который автоматически выключает его через заданный интервал времени.

После остановки насоса система вводит временную блокировку его повторного включения. Это делается для предотвращения частых включений-выключений, которые могут негативно сказаться на сроке службы оборудования. Продолжительность блокировки задана в алгоритме и позволяет почве впитать воду, прежде чем система произведёт следующую проверку. Во время блокировки насос остаётся отключённым, даже если данные с датчиков указывают на недостаток влаги.

В случае, если оба датчика были признаны неисправными или их данные вызывают серьёзные сомнения, насос немедленно отключается, чтобы избежать неконтролируемой работы. При этом система сигнализирует о проблеме, отображая соответствующее сообщение на дисплее.

Если один из датчиков работает корректно, а второй находится в неисправном состоянии, управление насосом осуществляется на основе данных только с рабочего датчика. Это позволяет системе продолжать функционировать в ограниченном режиме до устранения неисправности.

Работа пятого блока тесно связана с предыдущими этапами алгоритма, формируя замкнутый цикл. После завершения одного цикла работы насос снова переходит в режим ожидания новых данных от датчиков, начиная процесс управления заново. Такой подход обеспечивает гибкость и устойчивость системы, делая её надёжной даже в условиях нестандартных ситуаций.

* 1. **Описание блок-схемы алгоритма программы**

Для описания блок-схемы алгоритма автоматической системы полива определим последовательность операций, выполняемых устройством для мониторинга влажности почвы, управления насосом и обработки ошибок. Блок-схема структурирует основные этапы алгоритма, упрощая анализ логики работы и обеспечивая понимание функциональности системы. Блок-схема алгоритма, реализующего программное средство, представлена в приложении Г

Описание блок-схемы алгоритма системы автоматического полива:

1. Инициализация системы

После подачи питания микроконтроллер активирует компоненты системы: дисплей *LCD*1602 с *I*2*C*-адаптером, водяной мини-насос *DC5V 120L-H*, датчики влажности почвы *YL-69*. На этапе инициализации происходит настройка переменных (состояние насоса, параметры сенсоров) и базовых параметров работы (интервалы обновления, пороги влажности).

1. Считывание значений с датчиков

Для считывания используются два датчика влажности почвы, которые подключены к микроконтроллеру. Каждый датчик измеряет уровень влажности в своем участке почвы и передает аналоговые сигналы, которые преобразуются в цифровые значения. После получения сигналов датчики передают данные в микроконтроллер, где они интерпретируются как числовые значения в определенном диапазоне. Этот диапазон позволяет определить, насколько влажная или сухая почва в данный момент. Значения, близкие к верхнему пределу диапазона могут указывать на сухую почву, в то время как значения, приближающиеся к нижнему пределу, сигнализируют о том, что почва насыщена водой.

Для предотвращения ошибок и сбоев система проверяет корректность данных, поступающих с датчиков. Это особенно важно, если датчики неисправны, извлечены из почвы или по каким-либо другим причинам передают некорректные значения. В случае обнаружения некорректных данных система отмечает такие значения как невалидные и не принимает их во внимание при дальнейшей работе системы, а именно при определении поведения насоса.

Параллельно с анализом данные о текущих показаниях датчиков могут выводиться на экран для мониторинга. Это помогает пользователю убедиться в том, что система работает правильно, или вовремя заметить проблемы с датчиками. Если датчик передает некорректные данные, на экране отображается сообщение об ошибке, чтобы пользователь мог принять меры по устранению неисправности.

Считывание данных выполняется регулярно через определенные промежутки времени. Такой подход позволяет системе непрерывно отслеживать изменения уровня влажности почвы и оперативно реагировать на любые отклонения.

1. Вывод значений на дисплей

Вывод значений на экран выполняет информативную функцию, предоставляя пользователю данные о текущем состоянии системы и ее элементов. На дисплее с двумя строками отображается информация, которая помогает контролировать процесс и оперативно реагировать на возможные ошибки или отклонения.

На первой строке экрана выводятся текущие показания датчиков влажности почвы. Для каждого датчика указывается его идентификатор (*S*1 для первого датчика и *S*2 для второго) и измеренное значение. Эти значения представлены в числовом формате, где минимальные и максимальные границы определяют, насколько влажная почва. Диапазон датчика составляет от 0 до 1023, где значения ближе к 0 указывают на насыщенную влагой почву, а значения ближе к 1023 – на пересушенную. Такое отображение позволяет пользователю быстро оценить состояние почвы для каждого участка, обслуживаемого системой.

Вторая строка дисплея используется для отображения состояния системы и возможных ошибок. Такая опция помогает пользователю своевременно обнаружить неисправность и принять меры для ее устранения.

Во время работы насоса дисплей отключается. Это сделано для экономии энергии, поскольку насос потребляет значительное количество ресурсов, а питание дисплея в этот момент не является критически необходимым. Кроме того, отключение дисплея минимизирует отвлекающие факторы, что особенно актуально, если система работает автономно. Как только насос завершает свою работу, дисплей автоматически включается, чтобы пользователь мог снова получить доступ к информации о состоянии системы.

1. Анализ данных с датчиков

Анализ значений с датчиков и вывод информации об их состоянии является важным этапом работы системы, обеспечивающим проверку корректности данных, анализ уровня влажности и выявление возможных ошибок. На основе этих данных система принимает решения о дальнейшем управлении насосом и информирует пользователя о текущем состоянии.

Каждый датчик влажности передает числовое значение, отражающее уровень влажности почвы. Значения ближе к 0 указывают на высокий уровень влажности, а ближе к 1000 — на пересушенную почву. Для анализа данных система проверяет, попадают ли показания в установленный диапазон (например, от 100 до 900). Если показание выходит за пределы этого диапазона, датчик считается неисправным, и его данные игнорируются при дальнейшей обработке. Графический интерфейс пользователя, на котором продемонстрированы отклики устройства на различные типы ошибок и его состояние в стабильном режиме, представлен в приложении Д.

Система анализирует следующие типы ошибок:

*Err*: *Sensor* 1 (рисунок Д.1) – возникает, если показание первого датчика выходит за пределы допустимого диапазона (например, меньше 100 или больше 900). Система помечает первый датчик как неисправный, исключая его данные из дальнейшего анализа. На экране появляется сообщение «*Err*: *Sensor* 1», чтобы пользователь мог идентифицировать проблему. Управление насосом осуществляется на основании показаний второго, исправного датчика.

*Err*: *Sensor* 2 (рисунок Д.2) – аналогично, если второй датчик передает значения вне допустимого диапазона, он определяется как неисправный. Сообщение «*Err*: *Sensor* 2» отображается на дисплее для уведомления пользователя, а работа системы продолжается с учетом показаний первого датчика.

*Err*: *Both* *Sensors* (рисунок Д.3) – возникает, если оба датчика одновременно передают недопустимые значения. Система определяет оба датчика как неисправные, отключает насос, чтобы избежать некорректной работы, и выводит на экран сообщение «*Err*: *Both* *Sensors*». Пользователю рекомендуется проверить датчики и заменить их при необходимости.

Если оба датчика работают корректно, на экране отображается сообщение «*All* *Stable*» (рисунок Д.4) и значения датчиков в пределах допустимого диапазона. Система вычисляет среднее значение их показаний для повышения точности анализа уровня влажности. Сообщение «*All* *Stable*» на экране подтверждает нормальное функционирование системы без ошибок.

1. Работа насоса

Работа насоса напрямую зависит от анализа данных с датчиков влажности и текущих состояний системы. Основная задача алгоритма — включение насоса при недостаточной влажности почвы и его отключение после достижения приемлемого уровня.

Если показания датчиков влажности находятся ближе к 800 (примерное значение, указывающее на низкую влажность почвы), система определяет это как необходимость полива. Насос работает до тех пор, пока среднее значение показаний датчиков не уменьшится до уровня, указывающего на достаточную влажность (показания ближе к 400-500, что также является примерным значением, которое будет меняться в зависимости от потребностей растения к влажности почвы).

В случае отказа одного датчика система использует данные исправного сенсора. Насос продолжает функционировать в штатном режиме, полагаясь на корректные данные. Если оба датчика выходят за пределы допустимого диапазона или определены как неисправные, насос отключается для предотвращения непредвиденных ситуаций. На экране появляется предупреждение, и полив временно прекращается.

Для предотвращения перегрузки насоса и исключения частых включений/выключений применяется временная задержка. После каждого цикла полива насос переходит в режим ожидания на заданный интервал времени, чтобы оценить изменения влажности почвы. Это позволяет системе корректно учитывать динамику впитывания воды в грунт и избегать избыточного полива.

Если после завершения цикла полива влажность почвы остается низкой, система автоматически запускает новый цикл работы насоса. Однако между включениями поддерживается минимальный интервал времени для защиты оборудования.

1. Цикличное обновление

Алгоритм повторяется, пока устройство включено. После каждой итерации система возвращается к этапу считывания данных, обновляя значения сенсоров и управляя насосом на основании текущих условий.

Блок-схема описывает последовательность операций и принятие решений на каждом этапе, включая обработку ошибок, управление состояниями насоса и цикл обновления данных. Алгоритм обеспечивает автоматизацию полива растений, минимизируя вмешательство пользователя.

# ЗАКЛЮЧЕНИЕ

В рамках курсового проекта была разработана автоматическая система полива для комнатных растений на основе платформы *Arduino* *Nano*. Система успешно решает задачу поддержания оптимального уровня влажности почвы, минимизируя влияние человеческого фактора и обеспечивая стабильный уход за растениями в условиях ограниченного времени или отсутствия владельца.

В ходе работы проведено детальное исследование архитектуры вычислительной платформы *Arduino* *Nano* и её возможностей по взаимодействию с периферийными устройствами. Были изучены особенности работы датчиков влажности *YL*-69, выявлены их ограничения и сформированы способы отслеживания неисправностей при работе данных датчиков. На основе полученных результатов разработаны алгоритмы анализа данных, которые учитывают как штатные, так и аварийные ситуации, такие как отказ одного или обоих датчиков. Реализовано программное обеспечение, которое интегрирует сбор, обработку и визуализацию данных, а также управление исполнительными механизмами.

Система продемонстрировала высокую надёжность и устойчивость в различных режимах работы, включая реакции на краевые случаи и неисправности. Проведённый анализ и тестирование подтверждают её эффективность в решении поставленных задач.

Исходя из всего вышеперечисленного, можно сделать вывод о том, что проект полностью соответствует требованиям и задачам курсовой работы. Он направлен на повышение удобства ухода за комнатными растениями и оптимизацию использования воды. Полученные результаты имеют практическую значимость и могут быть использованы для дальнейшего совершенствования системы автоматизации ухода за растениями.

# СПИСОК ЛИТЕРАТУРНЫХ ИСТОЧНИКОВ

[1] Доманов, А. Т. Стандарт предприятия / А. Т. Доманов, Н. И. Сорока. – Минск: БГУИР, 2024. – 178 с.

[2] Гайд для новичков Arduino [Электронный ресурс]. – Режим доступа: *https://alexgyver.ru/arduino-first/.* – Дата доступа 21.09.2024.

[3] The History and Evolution of Arduino [Электронный ресурс]. – Режим доступа: *https://medium.com/uniting-digital/the-history-and-evolution-of-arduino-7dac03a732a4. –* Дата доступа: 21.09.2024.

[4] Архитектура *AVR* микроконтроллеров [Электронный ресурс]. – Режим доступа: *https*://*ph*0*en*1*x*.*net*/69-*avr*-*microcontroller*-*architecture*-*what*-*is*-*inside*-*avr*-*microcontroller*.*html*. – Дата доступа: 25.09.2024.

[5] Устройство микроконтроллеров *AVR* [Электронный ресурс]. – Режим доступа*: https://myrobot.ru/*stepbystep*/mcarchitecture.php.* – Дата доступа: 09.10.2024.

[6] Микроконтроллеры AVR. Лабораторный практикум : учеб. пособие / И. И. Кочегаров, В. А. Трусов. – Пенза : Изд-во ПГУ, 2012. – 122 с.

[7] Пьявченко, А. О. Архитектура, основы программирования и применения AVR-микроконтроллеров и ARM-микросистем. Часть 1 : учебное пособие / А. О. Пьявченко, В. А. Переверзев ; Южный федеральный университет. – Ростов-на-Дону ; Таганрог : Издательство Южного федерального университета, 2019. – 374 с

[8] *Guide* *for* *Soil* *Moisture* *Sensor* *YL*-69 *or* *HL*-69 *with* *Arduino* [Электронный ресурс]. – Режим доступа: *https*://*randomnerdtutorials*.*com*/*guide*-*for*-*soil*-*moisture*-*sensor*-*yl*-69-*or*-*hl*-69-*with*-*the*-*arduino*. – Дата доступа:  11.10.2024.

[9] *Arduino Nano, Pump* [Электронный ресурс]. – Режим доступа: *https://newbiely.com/tutorials/arduino-nano/arduino-nano-pump.* – Дата доступа:  11.10.2024.

[10] Миниатюрный водяной насос-помпа *RS-360SH*» ­– [Электронный ресурс]. – Режим доступа: *https://3d-diy.ru/blog/miniatyurnyj-vodyanoj-nasos-pompa-rs-360sh.* – Дата доступа: 12.10.2024.

[11] *Arduino Water Sensor Pump* [Электронный ресурс]. – Режим доступа: *https://arduinogetstarted.com/tutorials/arduino-water-sensor-pump.* – Дата доступа: 12.10.2024.

[12] Как подключить дисплей к Arduino [Электронный ресурс]. – Режим доступа: *https://oarduino.ru/voprosi-i-otveti/podklyuchenie-lcd-displeya-k-arduino-s-programmirovaniem*. – Дата доступа: 15.10.2024.

[13] *Arduino. LCD 1602. I2C* и другие варианты подключения [Электронный ресурс]. – *https://microtechnics.ru/arduino-lcd-1602-i2c-i-drugie-varianty-podklyucheniya.* – Дата доступа: 17.10.2024.

[14] *JQC-3FF-S-Z 5VDC 15A 5V Relay* [Электронный ресурс]. – Режим доступа: *https://en.tonglingrelay.com/product/JQC-3FF-S-Z-5VDC-15A-5V-Relay-1065.html?productCateId=76.* – Дата доступа: 01.11.2024.

[15] От 1.0 до 10: история развития *Microsoft* *Windows* [Электронный ресурс]. – Режим доступа: *https*://*tproger*.*ru*/*translations*/*microsoft*-*windows*-*history*. – Дата доступа: 14.11.2024.

[16] *Arduino* *Documentation* [Электронный ресурс]. – Режим доступа: *https*://*docs*.*arduino*.*cc*. – Дата доступа: 14.11.2024.

[17] *Arduino IDE Documentation* [Электронный ресурс]. – Режим доступа: *https://docs.arduino.cc/software/ide.* – Дата доступа: 15.11.2024.

[18] Капельный полив: зачем нужен, какой бывает и как подключить – [Электронный ресурс]. – Режим доступа: *https://www.sima-land.ru/o-kompanii/novosti-kompanii/5805.* – Дата доступа: 01.12.2024.

[19] История возникновения языка программирования *С++* [Электронный ресурс]. – Режим доступа: *http*://*cppstudio*.*com*/*post*/1984. – Дата доступа: 01.12.2024.

[20] Автополив комнатного цветка на Arduino – [Электронный ресурс]. – Режим доступа: *https://habr.com/ru/articles/210814.* – Дата доступа: 01.12.2024.

# ПРИЛОЖЕНИЕ А

# (обязательное)

# Справка о проверке на заимствования

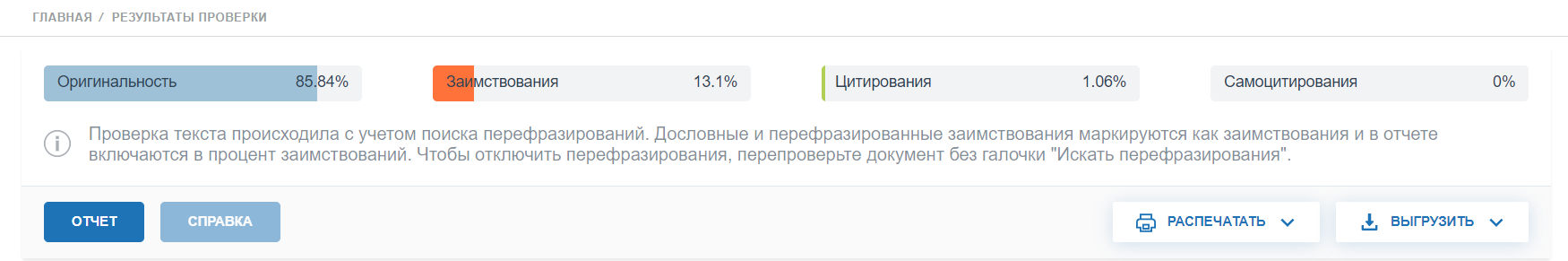


Рисунок 1 – Справка о проверке на заимствования

# ПРИЛОЖЕНИЕ Б

# (обязательное)

# Листинг программного кода

#*include* <*Wire*.*h*> // Библиотека для *I*2*C*

#*include* <*LiquidCrystal\_I*2*C*.*h*> // Подключение библиотеки для дисплея 1602

*const* *int* *sensorPin*1 = *A*0;  // Пин для первого датчика влажности

*const* *int* *sensorPin*2 = *A*1;  // Пин для второго датчика влажности

*const* *int* *pumpPin* = 2;       // Пин для насоса

*const* *int* *thresholdHigh* = 400; // Порог включения насоса

*const* *int* *maxSensorValue* = 900;

*const* *int* *minSensorValue* = 100;

*LiquidCrystal\_I*2*C* *lcd*(0*x*27, 16, 2);

*bool* *pumpOn* = *false*; // Состояние насоса

*bool* *sensor*1*Valid* = *true*;

*bool* *sensor*2*Valid* = *true*;

*unsigned* *long* *pumpStartTime* = 0; // Время включения насоса

*unsigned* *long* *pumpCooldownStartTime* = 0; // Время начала паузы после выключения

*unsigned* *long* *lastUpdateTime* = 0; // Время последнего обновления

*const* *unsigned* *long* *updateInterval* = 10000; // Интервал обновления измерений

*bool* *pumpCooldown* = *false*; // Флаг состояния блокировки насоса

*bool* *firstRun* = *true*; // Флаг для отслеживания первого запуска

*void* *setup*() {

*Serial*.*begin*(9600);

*pinMode*(*pumpPin*, *OUTPUT*);

*lcd*.*init*();

*lcd*.*backlight*();

}

*void* *loop*() {

*unsigned* *long* *currentMillis* = *millis*();

  // Обновление показаний датчиков по таймеру

*if* (*currentMillis* - *lastUpdateTime* >= *updateInterval*) {

*lastUpdateTime* = *currentMillis*;

    // Считывание значений с датчиков

*int* *sensorValue*1 = *analogRead*(*sensorPin*1);

*int* *sensorValue*2 = *analogRead*(*sensorPin*2);

    // Вывод значений датчиков на первой строке дисплея

*if* (!*pumpOn*) { // Дисплей включён только, если насос не работает

*lcd*.*setCursor*(0, 0);

*lcd*.*print*("*S*1:");

*lcd*.*print*(*sensorValue*1);

*lcd*.*print*(" *S*2:");

*lcd*.*print*(*sensorValue*2);

*lcd*.*print*("  ");

*Serial*.*println*("*Sensor* 1: ");

*Serial*.*println*(*sensorValue*1);

*Serial*.*println*("*Sensor* 2: ");

*Serial*.*println*(*sensorValue*2);

    }

    // Проверка значений и определение состояния датчиков

*checkSensors*(*sensorValue*1, *sensorValue*2);

    // Управление насосом в зависимости от состояния

*managePump*(*sensorValue*1, *sensorValue*2);

  }

  // Управление насосом при включенном состоянии (автоматическое выключение через *n* секунд)

*if* (*pumpOn* && (*millis*() - *pumpStartTime* >= 9000)) {

*digitalWrite*(*pumpPin*, *HIGH*); // Выключение насоса

*Serial*.*println*("*Pump* *off*");

*pumpOn* = *false*;

*pumpCooldown* = *true*; // Устанавливаем блокировку

*pumpCooldownStartTime* = *millis*(); // Запоминаем время начала блокировки

*lcd*.*backlight*(); // Включение подсветки дисплея

*updateDisplayStatus*(); // Обновляем статус на дисплее

  }

  // Снятие блокировки насоса через *n* секунд

*if* (*pumpCooldown* && (*millis*() - *pumpCooldownStartTime* >= 10000)) {

*pumpCooldown* = *false*; // Снимаем блокировку

  }

}

*void* *checkSensors*(*int* *sensorValue*1, *int* *sensorValue*2) {

*sensor*1*Valid* = (*sensorValue*1 <= *maxSensorValue* && *sensorValue*1 >= *minSensorValue*);

*sensor*2*Valid* = (*sensorValue*2 <= *maxSensorValue* && *sensorValue*2 >= *minSensorValue*);

  // Вывод ошибок на вторую строку дисплея, если насос не работает

*if* (!*pumpOn*) {

*if* (!*sensor*1*Valid* && !*sensor*2*Valid*) {

*lcd*.*setCursor*(0, 1);

*lcd*.*print*("*Err*: *Both* *Sensrs*");

    } *else* *if* (!*sensor*1*Valid*) {

*lcd*.*setCursor*(0, 1);

*lcd*.*print*("*Err*: *Sensor* 1   ");

    } *else* *if* (!*sensor*2*Valid*) {

*lcd*.*setCursor*(0, 1);

*lcd*.*print*("*Err*: *Sensor* 2   ");

    } *else* {

      // Если ошибок нет и насос не работает, показываем "Стабильно"

*lcd*.*setCursor*(0, 1);

*lcd*.*print*("*All* *Stable*      ");

    }

  }

}

*void* *managePump*(*int* *sensorValue*1, *int* *sensorValue*2) {

  // Если оба датчика находятся в ошибочном состоянии, отключаем насос и выходим

*if* (!*sensor*1*Valid* && !*sensor*2*Valid*) {

*digitalWrite*(*pumpPin*, *HIGH*); // Отключение насоса

*pumpOn* = *false*;

*pumpCooldown* = *false*; // Сбрасываем блокировку насоса

*return*;

  }

  // Если один из датчиков невалиден, управление насосом будет производиться только по валидному датчику

*if* (!*sensor*1*Valid* && *sensor*2*Valid*) {

*controlPump*(*sensorValue*2);

  } *else* *if* (*sensor*1*Valid* && !*sensor*2*Valid*) {

*controlPump*(*sensorValue*1);

  } *else* *if* (*sensor*1*Valid* && *sensor*2*Valid*) {

*int* *avgValue* = (*sensorValue*1 + *sensorValue*2) / 2;

*controlPump*(*avgValue*);

  } *else* {

    // В любом другом случае насос отключается

*digitalWrite*(*pumpPin*, *HIGH*);

*pumpOn* = *false*;

  }

}

*void* *controlPump*(*int* *sensorValue*) {

*if* (*firstRun*) {

*firstRun* = *false*; // После первого запуска сбрасываем флаг

*pumpOn* = *false*;

*digitalWrite*(*pumpPin*, *HIGH*);

*return*;

  }

  // Включение насоса при превышении порога и отсутствии блокировки

*if* (*sensorValue* > *thresholdHigh* && !*pumpOn* && !*pumpCooldown*) {

*digitalWrite*(*pumpPin*, *LOW*); // Включение насоса

*Serial*.*println*("*Watering* *is* *on*");

*pumpOn* = *true*;

*pumpStartTime* = *millis*(); // Запоминаем время включения

*lcd*.*noBacklight*(); // Отключение подсветки дисплея

  }

  // Выключение насоса через определённое время или по другому условию

*if* (*pumpOn* && (*millis*() - *pumpStartTime* >= 9000)) {

*digitalWrite*(*pumpPin*, *HIGH*); // Выключение насоса

*Serial*.*println*("*Watering* *is* *off*");

*pumpOn* = *false*;

*pumpCooldown* = *true*; // Устанавливаем блокировку

*pumpCooldownStartTime* = *millis*(); // Запоминаем время начала блокировки

*lcd*.*backlight*(); // Включение подсветки дисплея при выключении насоса

  }

}

*void* *updateDisplayStatus*() {

  // Обновление статуса на дисплее после выключения насоса

*if* (!*sensor*1*Valid* && !*sensor*2*Valid*) {

*lcd*.*setCursor*(0, 1);

*lcd*.*print*("*Err*: *Both* *Sensrs*");

  } *else* *if* (!*sensor*1*Valid*) {

*lcd*.*setCursor*(0, 1);

*lcd*.*print*("*Err*: *Sensor* 1   ");

  } *else* *if* (!*sensor*2*Valid*) {

*lcd*.*setCursor*(0, 1);

*lcd*.*print*("*Err*: *Sensor* 2   ");

  } *else* {

*lcd*.*setCursor*(0, 1);

*lcd*.*print*("*All* *Stable*      ");

  }

}

# ПРИЛОЖЕНИЕ В

# (обязательное)

# Функциональная схема алгоритма, реализующего программное средство

# ПРИЛОЖЕНИЕ Г

# (обязательное)

# Блок схема алгоритма, реализующего программное средство

# ПРИЛОЖЕНИЕ Д

# (обязательное)

# Графический интерфейс пользователя

# ПРИЛОЖЕНИЕ Е

# (обязательное)

# Ведомость курсового проекта